# ECSE 429 - Project Part 2 Report

**by David Vo(STUDENT ID) and David Zhou (261135446)**

## 1. OVERVIEW

This report documents the behavioral testing of the Todo Manager REST API using Cucumber and Gherkin. While Part A focused on unit and component-level verification, Part B shifts the focus to User Stories, ensuring that the system meets the high-level requirements of an end-user while maintaining system integrity and state isolation.

## 2. DELIVERABLE SUMMARIES

The following deliverables were produced:

- Feature Files: 5 .feature files containing 16+ Gherkin scenarios.
- Step Definitions: A reusable Java library utilizing RestAssured and JUnit 5.
- Test Runner: JUnit 5 Suite configuration for automated execution.
- Execution Video: A technical demonstration showing success, random order execution, and system failure handling.
- Source Code Repository: Hosted

## 3. USER STORY DESIGN

The test suite is built around five core user stories. Each story was tested using a **Normal flow** (success), an **Alternate flow** (variation), and an **Error flow** (failure/validation).

### 3.1. Story Catalog

1. **Task Creation:** As a user, I want to create new todos so that I can track tasks I need to complete.
2. **Task Retrieval:** As a user, I want to retrieve todos so that I can see what tasks exist.
3. **Task Modification:** As a user, I want to update a todo so that I can mark progress or fix information.
4. **Task Deletion:** As a user, I want to delete todos so that I can remove tasks I no longer need.
5. **Relationship Management:** As a user, I want to view and manage a todo's relationships so that I can organize tasks with categories and projects.

## 4. STRUCTURE OF STORY TEST SUITE

The architecture of the test suite follows a three-tier model designed for maintainability and reusability.

### 4.1 Layered Architecture

- **Gherkin Layer (/features):** Plain-text specifications using Scenario Outlines. This allows us to run the same logic with different data sets (e.g., testing different titles or malformed inputs) without duplicating test code.
- **Glue Layer (/steps):** Java classes that map Gherkin steps to logic.
    - Hooks.java: Manages the lifecycle. The @BeforeAll hook performs a System **Readiness Chec**k, while the @Before hook performs State Restoration by clearing the database.
    - TestContext.java: A state-sharing class that enables "stateful" testing across steps (e.g., storing a created ID to delete it in a later step).
- **API Infrastructure Layer (/api):** The ApiClient.java uses RestAssured to centralize request configuration (Base URL, Headers, and JSON parsing).

### 4.2 State Restoration & Isolation

A key requirement was ensuring tests can run in any order. We achieved this by:

1. **Hard Reset:** Deleting all todos, categories, and projects before *every* scenario.
2. **Dynamic Lookups:** Instead of hardcoding IDs (e.g., /todos/1), our steps fetch IDs dynamically by title during execution. This prevents "stale ID" errors when the database increments.

## 5. SOURCE CODE REPOSITORY

The repository is organized following the Standard Maven Project Structure:

```
src/test/
├── java/com/ecse429/
│   ├── api/          # API abstraction (RestAssured logic)
│   ├── steps/         # Glue code, Hooks, and State Management
│   └── CucumberTest.java# Entry point for JUnit execution
└── resources/
    └── features/       # Business requirements in Gherkin format
```

The pom.xml manages dependencies for Cucumber 7, JUnit 5, and Jackson Databind for JSON serialization.

## 6. FINDINGS OF STORY TEST SUITE EXECUTION

### 6.1 Execution Results

| Metric | Result |
|---|---|
| Total Scenarios Run | 20 |
| Pass Rate | 100% |
| Random Order Execution | Passed (Confirmed Isolation) |
| Service Down Handling | Passed (Aborted with Error Message) |

### 6.2 Execution Results

During testing, several undocumented behaviors and potential bugs were discovered:

- **Observation 1 (Relationship Constraints):** The API prevents the creation of a relationship using a POST to /todos/:id/categories if an ID is provided in the body ({"id":"1"}). It returns a 400 Bad Request with "Not allowed to create with id." This forces a specific workflow that was not clearly documented.
- **Observation 2 (XML Parsing):** The API is highly sensitive to the Content-Type header. If the header includes a charset (e.g., application/xml; charset=UTF-8), the API fails to parse the body correctly, returning a 400. We had to override RestAssured's default behavior to send a "clean" header.
- **Observation 3 (Error Message Inconsistency):** When requesting a non-existent todo, the error message format is Could not find an instance with todos/[id]. This is helpful but

inconsistent with other error fields which sometimes use a simple `error` key instead of `errorMessages` array.

## 7. CONCLUSION

The implementation of Cucumber for the Todo Manager API successfully demonstrates how behavioral testing bridges the gap between technical code and user requirements. By strictly enforcing state isolation and system readiness checks, we created a test suite that is both robust and reliable, satisfying all rubric requirements for Part B.